Patent Application of
Pilla Gurumurty Patrudu
for

TITLE : PARALLEL PROCESSING SYSTEM DESIGN AND ARCHITECTURE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is entitled to the benefit of Provisional Patent Application Number 60/183,660 filed 2000 Feb 18.

BACKGROUND—FIELD OF INVENTION

This invention relates to automating the development of multithreaded applications for computing machines equipped with multiple symmetric processors and shared memory.

BACKGROUND—DESCRIPTION OF PRIOR ART

Multithreaded applications for symmetrical multiprocessor (SMP) machines, require a significant amount of synchronization code to work properly.

The developers are required to develop the synchronizing code, by using the primitive synchronization constructs like spin locks, event objects, semaphores, and critical sections. These primitive constructs are provided by the host language, operating system or by third party libraries.

Developers trying to harness the power of multiple processors of a SMP machine, often find that the synchronization code, is more complex, than the applications which they are developing, since the synchronizing code demands professional computing skills. In other words developers attempting to harness the power of SMP machines had to transform themselves as psuedo computer scientists.

Some programming languages like Java, provide language constructs to simplify the synchronization. However these language constructs are still far away from abstracting the synchronization requirements, and still require significant coding and understanding of the synchronizing mechanisms. One of the major problems with the java language constructs for synchronizing is that they are too naïve and may incur significant performance loss in some applications, unless intelligent objects or components are built using the basic language constructs. Again the developer has to acquire professional computing skills to harness the power of the SMP machines.

Despite the primitive synchronization constructs provided by the operating systems, and the language constructs, developers still have to work around dead locks, since there are no known constructs which provide transactional locking, that is, a mechanism by which a group of resources may all be acquired, or none of them acquired.

Due to the excessive complexity of developing synchronizing code, parallel computing using SMP machines is still relatively unexploited, despite the cheap prices of the systems.

Data flow computing is an alternative to thread based computing, however data flow computing mechanisms are implemented in hardware, and the machines are called data flow computers. Data flow computing constructs are executed in parallel, and the synchronization requirements are automatically detected and implemented by the hardware. These machines are quite expensive and are still not found in widespread commercial use.

SUMMARY

The present invention is based on the key features of the data flow architecture, and the threading model for parallel computing, and the resulting hybrid architecture is named "Resource Control Programming (RCP)" architecture. Rcp is a software architecture, which utilizes coarse grained scheduling and data flow computing architecture. Applications implementing the Rcp architecture, can make use of Rcp runtime modules and Rcp runtime libraries. The Rcp runtime libraries provide extensive run time support, for most of the synchronization requirements of the application.

OBJECTS AND ADVANTAGES

The most important objects and advantages of the present invention, are :

a) The management of inputs/outputs and functions, is undertaken by the Rcp runtime library, and the developer is relieved from the trouble of coding the complex synchronization code.

b) The Rcp runtime library automatically detects and balances the load, which is a great boon for developers, since load balancing is a very complex issue in parallel computing.

Further objects and advantages of the present invention are :

a) The threading model is abstracted by the Rcp architecture, and the Rcp runtime creates and manages the threads and performs controlled termination at the end of the application.
b) The application design and development are clearly segregated so that a person with greater knowledge of the application can design the application, and developers with more knowledge of programming languages can develop the application.
c) The application design is independent of the host language, operating system, and can be ported to other host languages or operating systems without any changes.

DRAWING FIGURES

The present invention will be described with reference to the accompanying drawings, wherein :

Figure 1 is a block diagram illustrating the Rcp runtime library

Figure 2 is a block diagram illustrating the schematic of a Queue

Figure 3 is a block diagram illustrating the schematic of a Queue Array

Figure 4 is a block diagram illustrating the schematic of a Virtual Queue

Figure 5 is a block diagram illustrating the schematic of a Node function

Figure 6 is a block diagram illustrating the schematic of a Rcp Gate

Figure 7 is a block diagram illustrating the Rcp Gate Interconnections

Figure 8 is a block diagram illustrating the Rcp Translator

Figure 9 is a block diagram illustrating the schematic of Rcp Load Image File layout

Figure 10 is a block diagram illustrating the schematic of Rcp Load Image Header Structure

Figure 11 is a block diagram illustrating the schematic of a Frame Structure

Figure 12 is a block diagram illustrating the schematic of a Worker structure

Figure 13 is a block diagram illustrating the Rcp runtime Library

Figure 14 is a block diagram illustrating the schematic of a Run_id structure

Figure 15 is a block diagram illustrating the schematic of a Queue structure

Figure 16 is a block diagram illustrating the schematic of a Queue Info structure

Figure 17 is a block diagram illustrating the schematic of a Node function structure

Figure 18 is a block diagram illustrating the schematic of a Node Function Info structure

Figure 19 is a block diagram illustrating the schematic of a Local Ring structure

Figure 20 is a block diagram illustrating the schematic of a Queue Status structure

Figure 21 is a block diagram illustrating the schematic of a Queue Data Node structure

Figure 22 is a block diagram illustrating the schematic of a Queue header structure

Figure 23 is a block diagram illustrating the schematic of a Lock Structure

Figure 24 is a block diagram illustrating the schematic of a Queue Array Node structure

Figure 25 is a block diagram illustrating the schematic of a Bind sequence number structure

Figure 26 is a block diagram illustrating the schematic of a Virtual Queue Node structure

Figure 27 is a block diagram illustrating the schematic of a Node function status structure

Figure 28 is a block diagram illustrating the schematic of a Rcp Gate Node structure

Figure 29 is a block diagram illustrating the schematic of a bind node structure

Figure 30 is a block diagram illustrating the schematic of a Node function Invocation structure

Figure 31 is a block diagram illustrating the schematic of a Rcp runtime Library

Figure 32 is a block diagram illustrating the schematic of a Rcp runtime Library

Figure 33 is a block diagram illustrating the Node function configuration of a Sample Application

Figure 34 is a block diagram illustrating the Rcp gate configuration of Sample Application

Figure 35 illustrates the Main Function of Sample application

Figure 36 illustrates the Claim Selector Function of Sample application

Figure 37 illustrates the Claim Processor Function of Sample application

Figure 38 illustrates the Reject Function of Sample application

Figure 39 illustrates the Payment Function of Sample application

Figure 40 illustrates the Tables of Sample Application

Figure 41 illustrates the trace of Sample Application

Figure 42 illustrates the trace of Sample Application

Figure 43 illustrates the trace of Sample Application

Figure 44 illustrates the trace of Sample Application

Figure 45 illustrates the trace of Sample Application

Figure 46 illustrates the trace of Sample Application

Figure 47 illustrates the trace of Sample Application

Figure 48 illustrates the trace of Sample Application